Инструментарий хакера: RU32.EFI

26 Фев 2016

Илья Манусов

Проектирование технических устройств и систем нередко связано с необходимостью учета различных взаимно-противоречивых требований. Софт-индустрия — не исключение из этой закономерности: перед разработчиками приложений стоит задача своевременной поддержки новых аппаратных платформ и операционных систем и в то же время — сохранения совместимости с устаревающей, а иногда и безнадежно устаревшей инфраструктурой.

На примере информационно-диагностической утилиты *RU32* рассмотрим один из подходов к решению такой задачи. Эта утилита существует в версиях для 16-битной среды *MS-DOS*, а также 32/64-битных реализаций *UEFIFirmware*. На сайте разработчика, к слову сказать — сотрудника American Megatrends, аккуратно анонсированы работы над ARM-версией этой утилиты. Небольшое исследование позволит оценить существующую «кроссплатформенность от AMI» в действии. Опыты выполнялись на виртуальных машинах в среде Oracle Virtual Box.

MS-DOS

В этом «ностальгическом» режиме, проблем совместимости обнаружено не было. Отметим, корректную динамическую ревизуализацию объектов, состояние которых изменяется во время просмотра. Речь о ячейке памяти в области переменных Legacy BIOS, по адресу 0000:046Ch, инкрементируемую по прерыванию от таймера с частотой около 18.2 Hz. В ответ на попытку просмотра переменных UEFI в среде Legacy BIOS, утилита визуализировала дамп памяти с адреса 0, не заявив о некорректности такого действия.

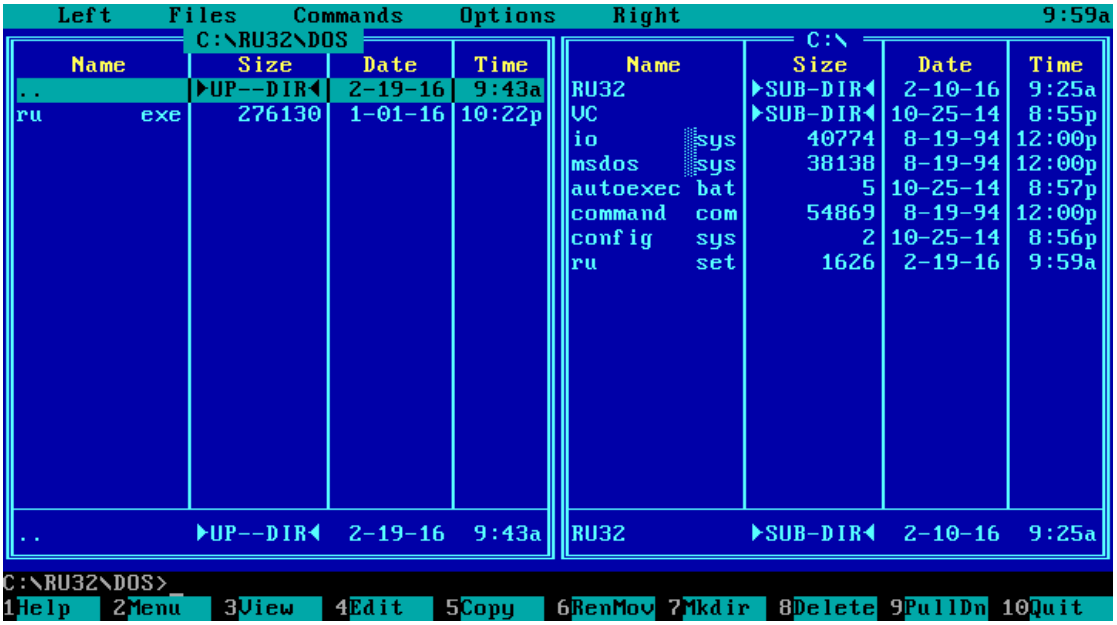

Рис.1. *На виртуальной машине загружена операционная система MS-DOS и Volkov Commander*

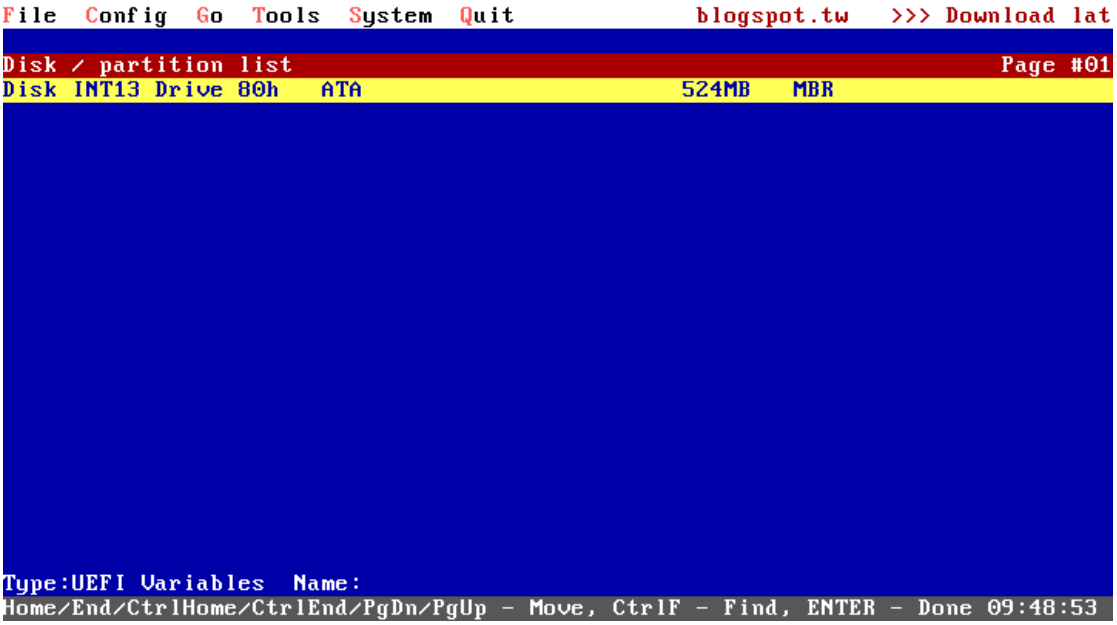

Рис.2. *Список устройств хранения данных (mass storage)*

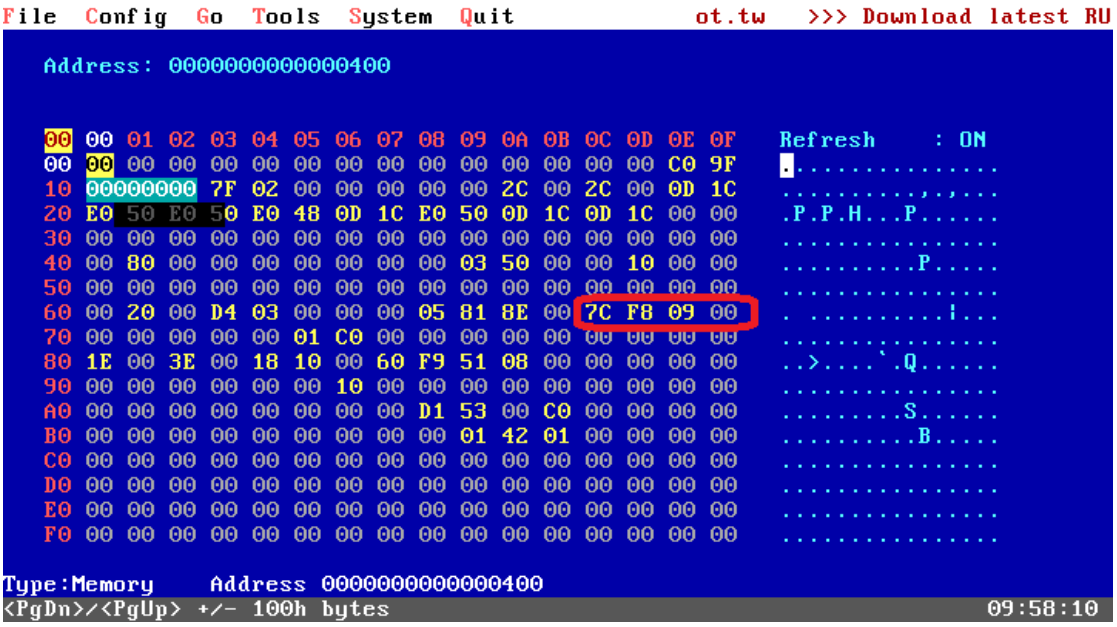

Рис.3. *Просмотр блока переменных Legacy BIOS*

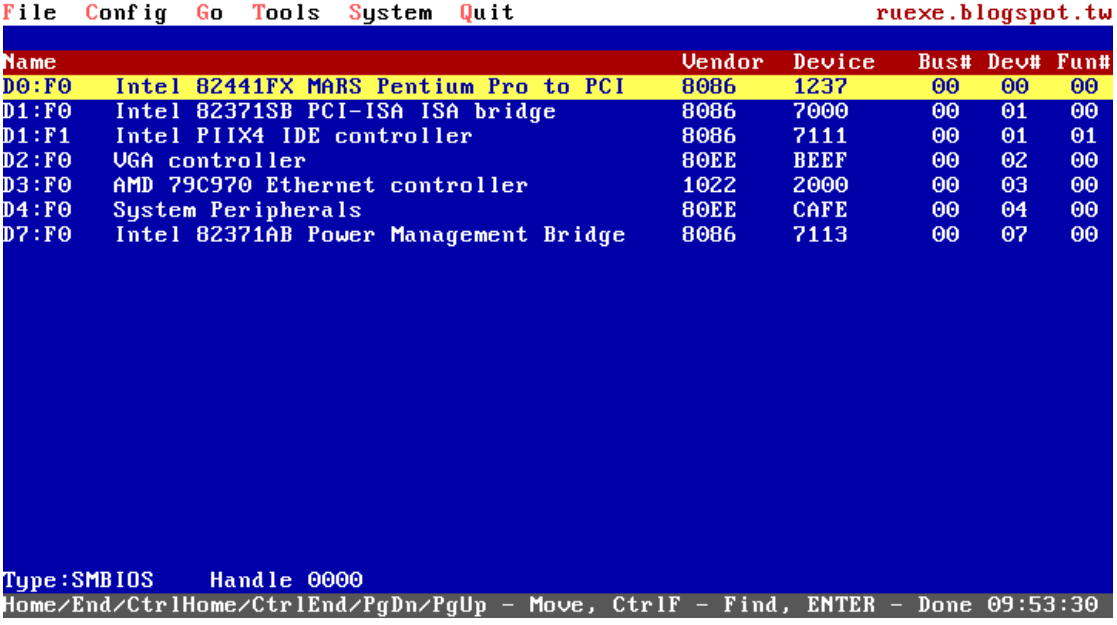

Рис.4. *Список PCI устройств*

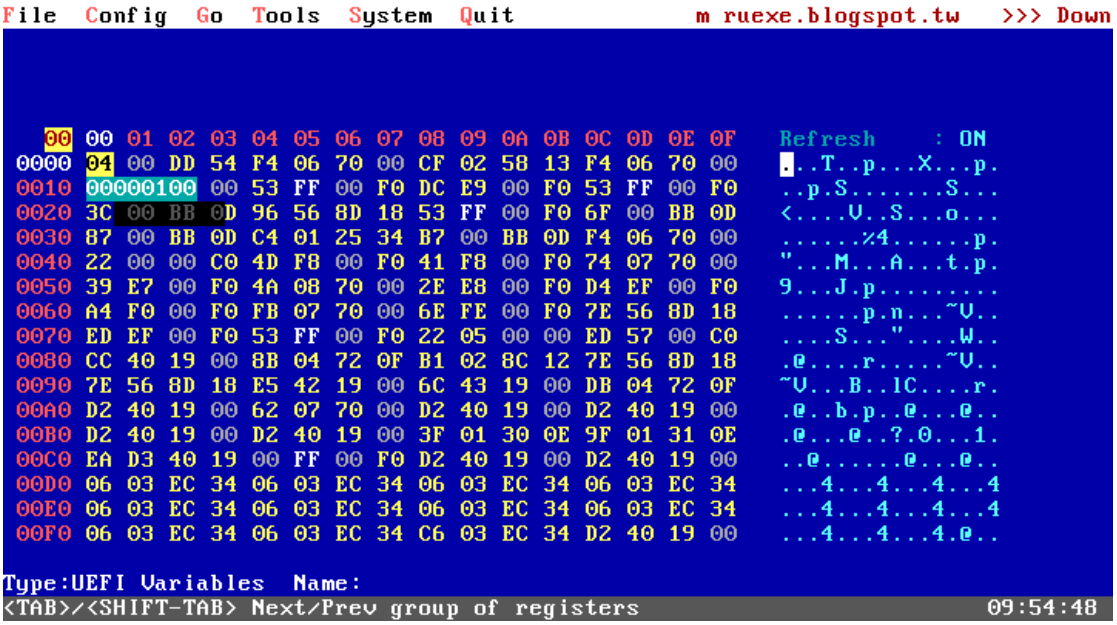

Рис.5. *Режим UEFI variables. В Legacy контексте, визуализируется область памяти с адреса 0*

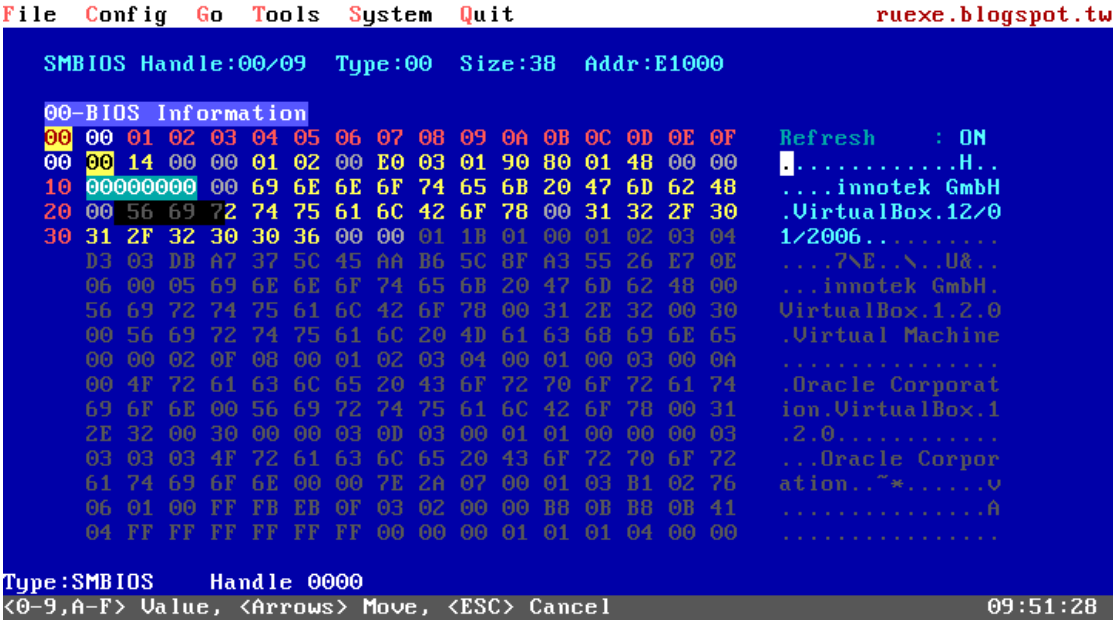

Рис.6. *Информация о BIOS виртуальной машины, предоставляемая функциями SMBIOS*

UEFI IA32

В этом режиме, для обеспечения работоспособности утилиты в среде *Oracle Virtual Box*, потребовалась настройка виртуальной среды, причем достаточно радикальная. Утилита «зависала» с темным экраном. В целях совместимости, пришлось отказаться от *аппаратной* виртуализации (см. рис.7). Как и следовало ожидать, производительность в режиме *программной* виртуализации значительно ниже, это заметно по плавному скроллингу в *UEFI Shell*. Вместе с тем, даже в режиме программной виртуализации, скорость обновления экрана в утилите *RU32* достаточно высокая, что говорит о хорошей оптимизации кода. *Отметим, что несовместимость замечена только при запуске конкретной версии RU32.EFI (RU5.16.0248) в сочетании с конкретной версией Oracle Virtual Box (версии 5.0.10 r104061).*

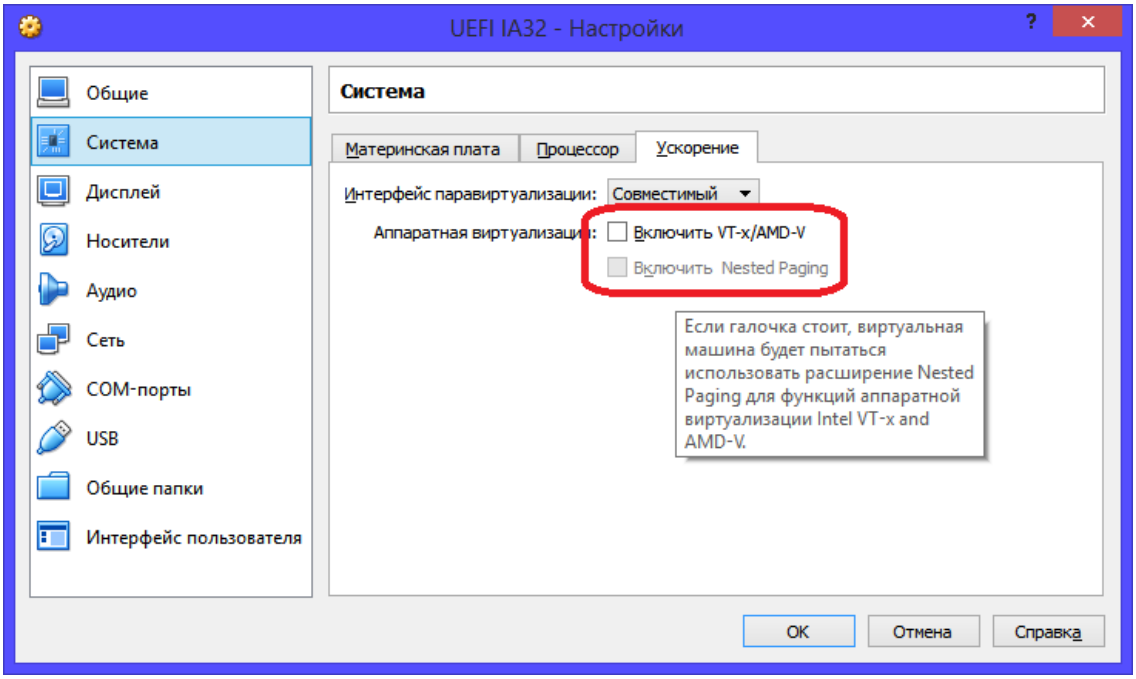

Рис.7. *Отключаем аппаратную виртуализацию VT-x и опцию управления страницами Nested Paging*

Настроив виртуальную машину, приступаем к исследованиям. Пытаясь увидеть переменные Legacy BIOS в режиме UEFI, получаем блок нулевых байтов (Рис.10). Отметим, что такой «разрыв с прошлым» имеет место не на всех платформах. Блок UEFI variables (рис.12) более гибок, как в смысле размещения переменных, так и с точки зрения их статуса. *NV* означает *Non Volatile* статус, *BS (Boot Services)* означает доступность переменной только на фазе загрузки ОС, до вызова системной функции *ExitBootServices()*. *RT (Runtime Services)* означает доступность переменной в сеансе ОС. Так как оболочка UEFI Shell не вызывает функцию *ExitBootServices()*, видимо не претендуя на статус полноценной операционной системы, переменные и API со статусом *BS*, доступны UEFI приложениям.

```
fs0:\> map
Device mapping table
  fs0   :HardDisk - Alias hd20b blk0
         Acpi(PNP0A03,0)/Pci(D10)/?/HD(Part1,SigEF09EF09)
  blk0  :HardDisk - Alias hd20b fs0
         Acpi(PNP0A03,0)/Pci(D10)/?/HD(Part1,SigEF09EF09)
  blk1  :HardDisk - Alias (null)
         Acpi(PNP0A03,0)/Pci(D10)/?/HD(Part2,SigEF09EF09)
  blk2  :HardDisk - Alias (null)
         Acpi(PNP0A03,0)/Pci(D10)/?/HD(Part2,SigEF09EF09)/HD(Part1,Sig00000000)
  blk3  :BlockDevice - Alias (null)
         Acpi(PNP0A03,0)/Pci(111)/Ata(Secondary,Master)
  blk4  :BlockDevice - Alias (null)
         Acpi(PNP0A03,0)/Pci(D10)/?
  hd20b :HardDisk - Alias fs0 blk0
         Acpi(PNP0A03,0)/Pci(D10)/?/HD(Part1,SigEF09EF09)

fs0:\> ver
EFI Specification Revision : 2.31
EFI Vendor                 : EDK II
EFI Revision               : 1.0
EFI Build Version          : 20061109

fs0:\>
```

Рис.8. *На виртуальной машине загружен UEFI Shell для IA32 UEFI*

Рис.9. *Список устройств хранения данных (mass storage)*

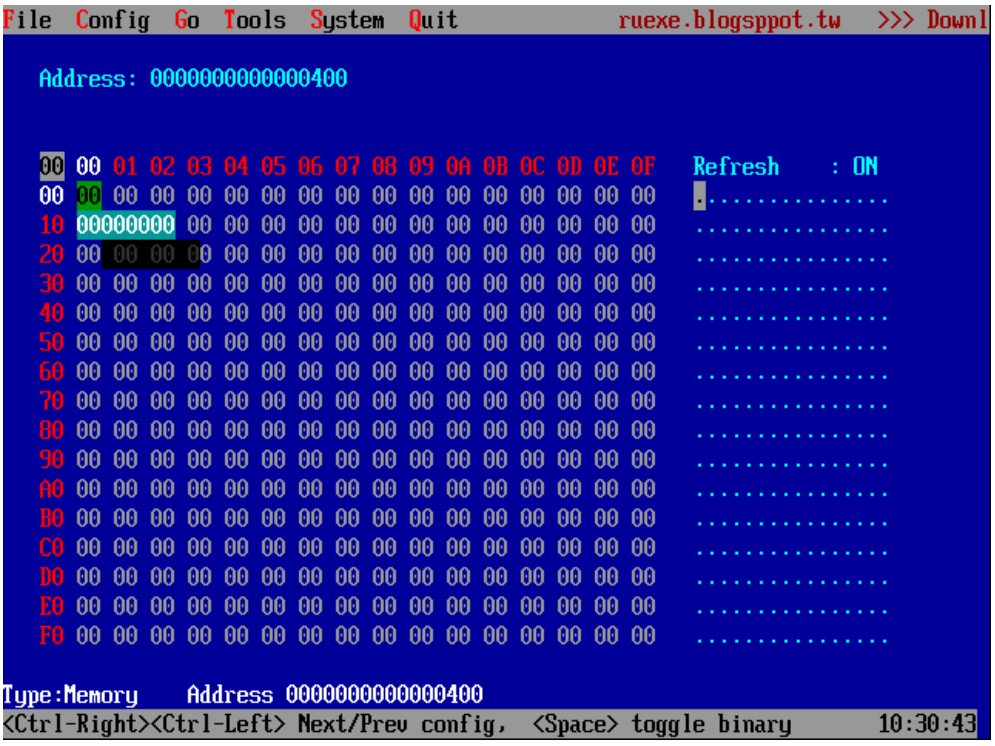

Рис.10. *Просмотр переменных Legacy BIOS по адресу 0000:0400h. В контексте UEFI этот блок не используется и обнулен*

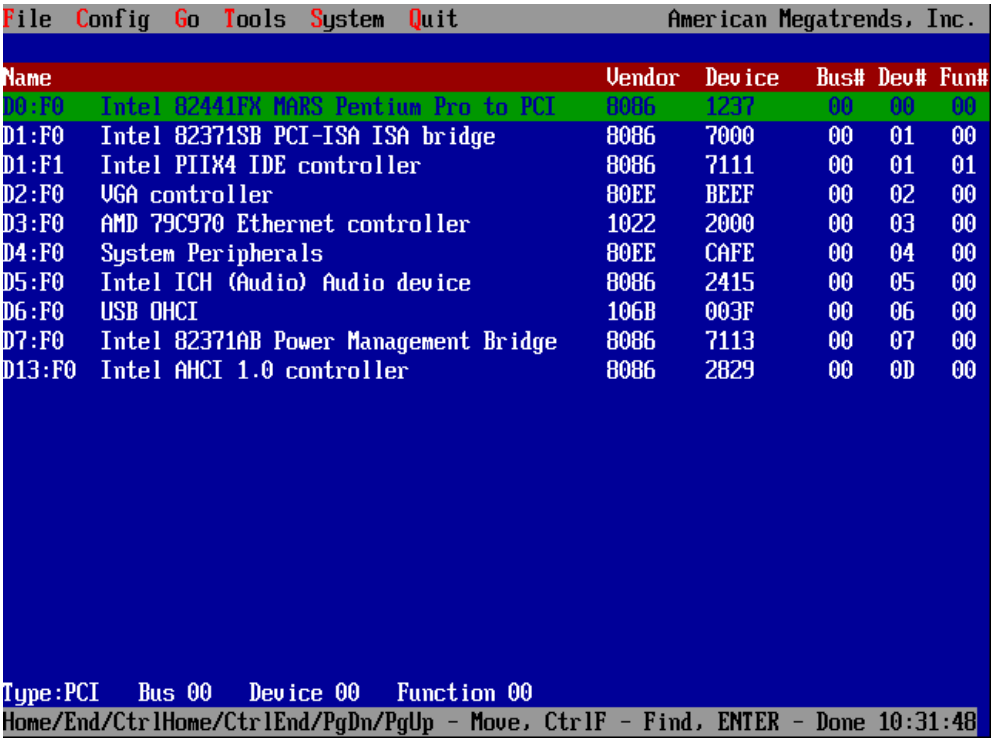

Рис.11. *Список PCI устройств*

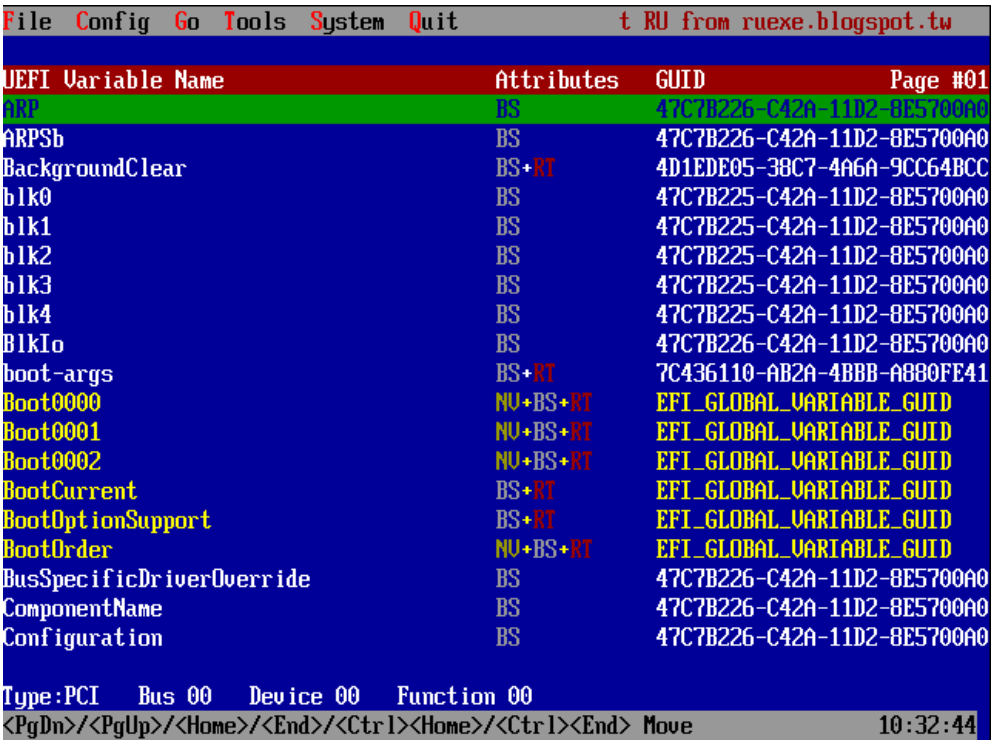

Рис.12. *Режим UEFI variables. Просмотр переменных UEFI Firmware*

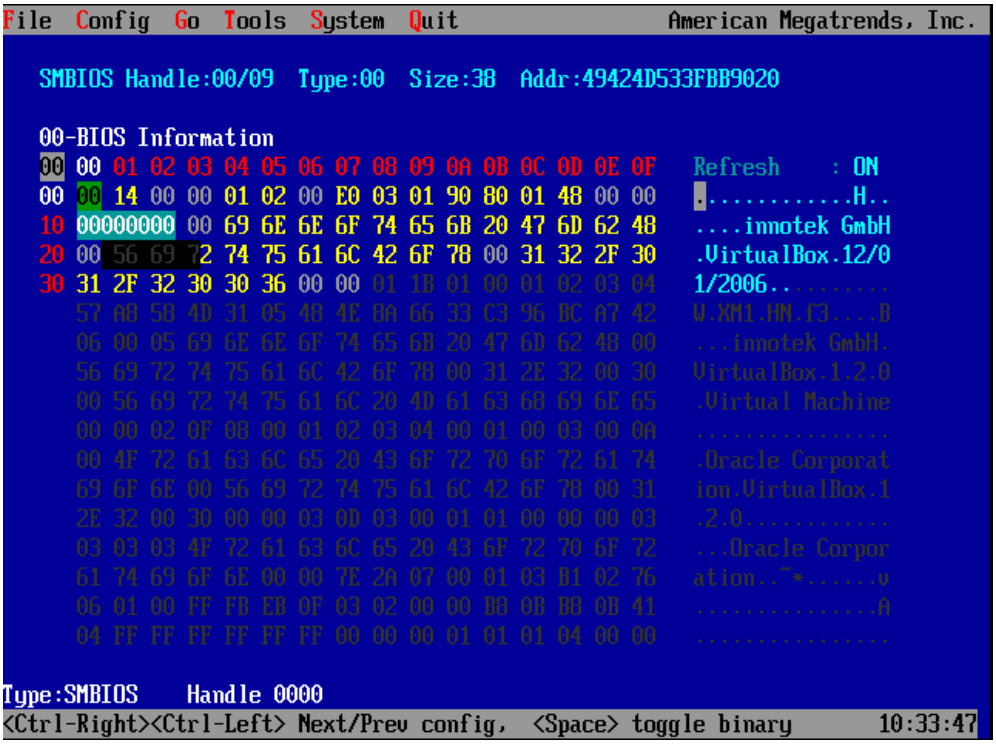

Рис.13. *Информация о BIOS виртуальной машины, предоставляемая функциями SMBIOS*

UEFI x64

Работоспособность утилиты RU32 в виртуальной среде UEFI x64, также зависит от установки опций Oracle Virtual Box. Но в отличие от 32-битного варианта, такие радикальные меры, как переход от аппаратной виртуализации к программной, здесь не потребовались. Ключевым моментом является тип южного моста системной логики. Если выбран относительно современный мост ICH9 включается поддержка архитектуры шины PCI Express на виртуальной платформе, при этом регистры конфигурационного пространства адресуются в адресном пространстве памяти, как *Memory Mapped I/O*. *Предположительно*, именно этот фактор вызывает конфликт при запуске RU32. Выбор менее современного моста PIIX3, позволяет запустить утилиту RU32 на виртуальной машине.

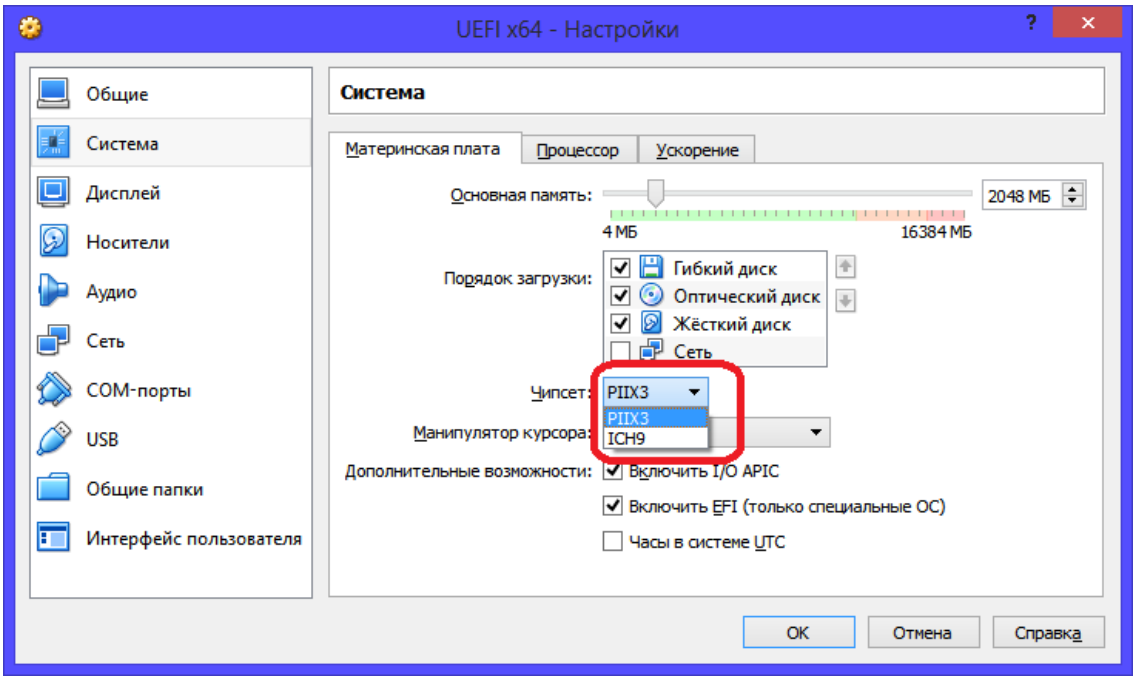

Рис.14. *Выбираем модель южного моста системной логики. Полезным побочным эффектом от установки режима PIIX3, является переход к классическому методу адресации конфигурационного пространства, при этом диапазон Memory Mapped Configuration (MCFG) на виртуальной платформе отсутствует*

Не будем приводить аналогичный набор скрин-шотов для UEFI x64, поскольку их содержимое практически совпадает со снимками, сделанными в режиме UEFI IA32. Вместо этого акцентируем внимание на ресурсах, состояние которых в 32 и 64-битном режиме различно. Рассмотрим дамп Model-Specific регистров процессора.

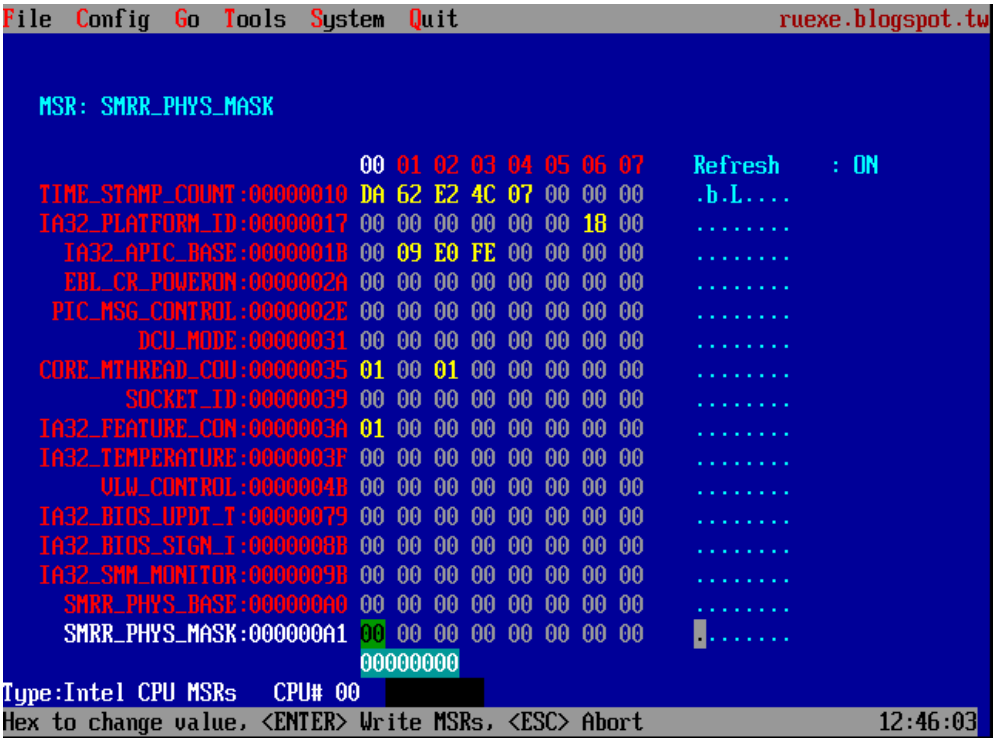

Рис.15. *Дамп Model-Specific регистров в режиме IA32 UEFI*

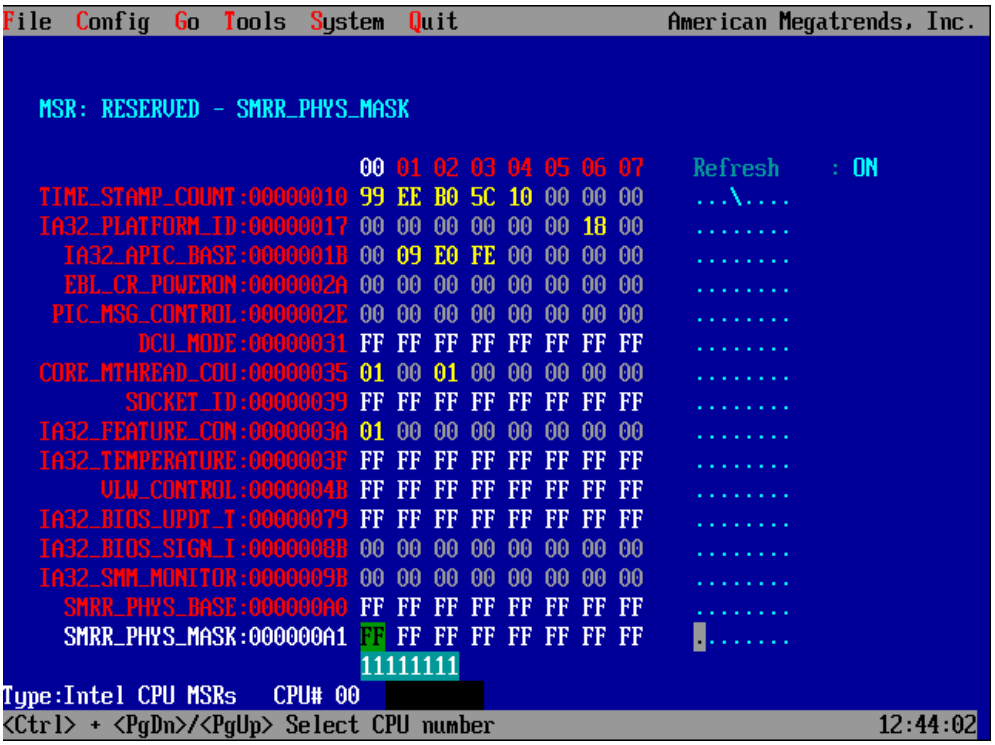

Рис.16. *Дамп Model-Specific регистров в режиме x64 UEFI*

Регистр *TIME_STAMP_COUNT* является счетчиком процессорных тактов, его состояние постоянно изменяется, различные значения в 32 и 64 битном варианте связаны только с этим фактом. Здесь мы еще раз убедились в способности RU32 адекватно визуализировать объекты, состояние которых изменяется во время просмотра. Обратим внимание на несколько регистров, содержимое которых в 32-битном режиме — нулевое, а в 64-битном — максимальное (FFFFFFFFFFFFFFFFh). Например, регистр термоконтроля *IA32_TEMPERATURE*. Предположительно, RU32, в случае недоступности регистра, визуализирует это значение. Отметим, что более информативно было бы сообщить о недоступности регистра явно (не ограничиваясь неочевидным

комментарием над дампом), иначе получаем некоторую неопределенность, ведь регистр может быть доступен и при этом содержать данное значение.

Вместо послесловия: цель оправдывает средства

Эффективный инструмент низкоуровневого исследования платформ по определению не может быть бесконфликтным. Утилита предназначена для профессионального применения, специалистами, которые знают, чего они хотят и что для этого надо сделать. Блокировать некоторые возможности, находящиеся на границе устойчивости, чтобы защитить хакеров от самих себя, было бы неверно.